

IMAGE PROCESSING AND AUTOMATIC SOLVING OF NONOGRAMS

Bálint Udvardy

Bachelor Degree Programme (3.), FEEC BUT

E-mail: xudvar00@stud.feec.vutbr.cz

Supervised by: Ilona Janáková

E-mail: janakova@feec.vutbr.cz

Abstract: In many puzzle books we come across logical puzzles like sudoku or nonograms (also known as griddlers or piccrosses), which demand mathematical logic to solve them. This article focuses on solving nonograms based on digital images of the puzzle. This solution is restricted to black and white nonograms, which can be slightly distorted by the angle of the camera during capture.

Keywords: nonograms, image processing, segmentation, puzzle solver

1 INTRODUCTION

Nowadays, most of the logical puzzles are also available in a form of mobile applications, in which the check of the result is quite easy, considering that the application knows the correct solution for the given puzzle. The goal of this work is to get the information from a picture taken on a smartphone or from a scanned document. The numbers around the central area of the puzzle are sufficient to calculate the solution if these numbers are correctly grouped. In addition, the central area could be recreated digitally, to display the solution or to give hints for the user, whether the selected area should or should not be filled.

All algorithms created to solve nonograms are solvers, which can solve these types of puzzles knowing the legend. I have not found any solution, which also deals with image processing. Because of that certain restrictions of possible input-images are needed: fixed legend location, etc.

2 PREPROCESSING

For reducing the amount of data for further processing firstly, the amount of colour components should be reduced. The average input image uses the RGB additive colour palette. The resulted greyscale image is then binarized using adaptive thresholding, which adjusts the value of each pixel according to its neighbourhood. It gives us a much better overall result than a global threshold would, as you can see in the images below.

3 SEGMENTATION

As nonograms have no defined aspect ratio, template matching cannot be used for finding the region of interest. We assume, that the picture was taken with its purpose in mind (e.g. to contain the picture of one nonogram, which the user wishes to solve), so the largest area is covered by the square grid. Firstly, Hough Transform is being used to find line com-

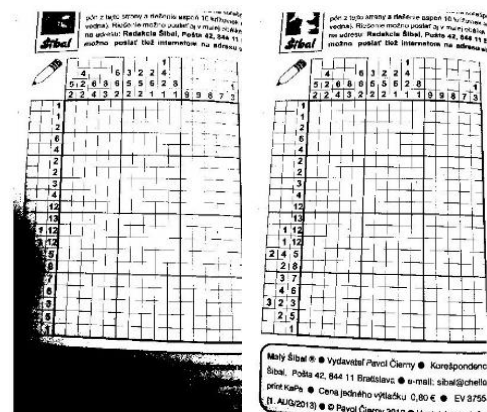


Figure 1: The result after global thresholding

Figure 1: The result of adaptive thresholding

ponents in the image (the most lines should be part of the puzzle grid). According to the dominant angle of these lines (e.g. the modus) the image is rotated expanding the output image if necessary, lest the corners should be chopped. By finding the bounding box of the largest connected component, which should be the grid, the image is cropped. At this point, the picture should contain only the puzzle with minimum amount of surrounding text or other unwanted objects.

3.1 GEOMETRICAL CORRECTION AND PICTURE ADJUSTMENT

Separating the largest connected component gives ample opportunity to define the average cells size. It could be defined by the intersections of houghlines perpendicular to each other. Because of the restricted usage of these lines in matlab I have chosen another method. From the negative of the image containing exclusively the puzzle grid matlab can the area of the cells rather facilely. Of course, the largest white area will be around the puzzle grid and there will be smaller areas (with the extension of 1 to 10 pixels). I used morphological opening on the image to remove (or at least reduce the number of) the previously mentioned areas. Using trimmed mean with a 30 per cent of trimming on the vector, which contains the number of pixels each area has, proved to be sufficient as further shrinking the data from which the average is being calculated, did not change the results overly. In the possession of the knowledge, that the grid cells are square-shaped, the square root of the previously calculated average gives a rough estimation of their size.

3.2 FURTHER SEGMENTATION

In the next section I aim to divide the already segmented image of the nonogram to 3 additional segments: two separate segments containing digits and one containing the part to be filled.

To separate the digits from the rest of the components, the same method I used to estimate the cell size of the grid can be used, only on the inverse image. By removing every component which is at least twice as big as the trimmed mean or less than the half of it, should leave us only the digits. The digit with the highest pixel count is usually 8 and the one containing the least amount of pixels is 1. Both are in the value range I set up earlier. It is worth mentioning, that some digits could about the puzzle grid thus removed with it. That requires further inspection of certain areas during the optical character recognition, but the division of the image is not affected by this error. To separate the digits above and the digits next to the grid the upper left corner is needed, the position of which can be calculated by a simple algorithm. I summed every row of the image into a vertical vector and every column into a horizontal vector. The gaps between digits are resembled by zeros in the vectors, the presence of the digits by the actual sum of active pixels in that row/column (it is a number greater than one usually). In some cases, finding the index of the maximum in each vector and adding/subtracting half of the grid cell size to/from them would give the coordinates to chop the image satisfactorily as seen on the picture below.

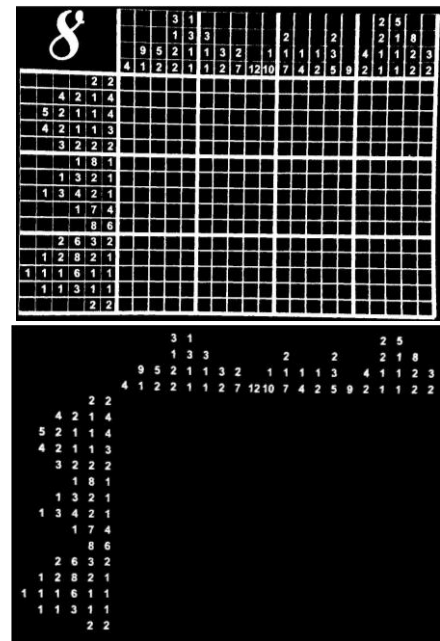


Figure 3: The image of a nonogram before (top picture) and after (bottom picture) the digit segmentation.

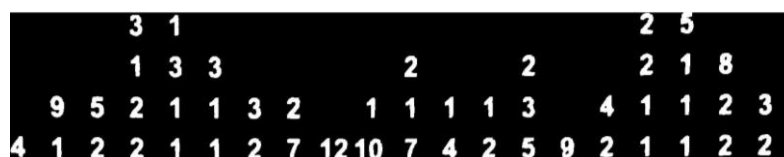


Figure 4: cropped image, containing only the numbers above the puzzle grid

This method does not always work, so I wrote my own function, which finds the coordinates of the last non-zero number in a cluster of numbers, the sum of which is the greatest and it is followed by another cluster of numbers, which have a significantly lower sum as the previous cluster. It works effectively even on nonograms where the bottom two rows from the group of numbers above the grid have a comparable amount of sum.

The rest of the image, probably containing the puzzle grid, can be separated easily from the rest of the image. Because of the thresholding the grid might be broken. To correct it, I used morphological closure on the image.

4 IDENTIFICATION AND SOLUTION

4.1 DIGIT RECOGNITION

On the first two images the optical character recognition (later used as OCR) of matlab can be used quite effectively. If the number consists of two digits it can be recognised by the distance of these digits, which can be easily determined from the bounding boxes the OCR function returns. From the coordinates it should be clear which row and column they belong to. It is a known fact, that on the image containing the numbers above the grid the bottom line contains the most numbers, in case of the second one, the left-most column. Again, if not, checking the expected positions on the previously mentioned image is essential. By sorting the digits and arranging the numbers into structures the most solvers demand, the puzzle can be solved.

4.2 RECOGNISING ALREADY FILLED REGIONS IN THE PUZZLE GRID

In the last image, containing supposedly only the puzzle grid, there should be no problem to find the intersection of gridlines with a cross-shaped-mask, with the size of two fifths of the average side of the grid cells and with expanding arm-width towards the edges of the mask. After finding the skeleton of the grid using morphological thinning (as the other methods, like Gao Hall and other parallel methods are not available as a core function in matlab) and applying 2D convolution with the mentioned mask on it we should get the positions of the grid line intersections. If there is a line intersection detected within the range of two thirds of the average side-size, it is removed.

Most of the nonogram solving algorithms need only the numbers, so the additional information derived from the third sub-image could be used to give users hints, whether their solution is correct or not.

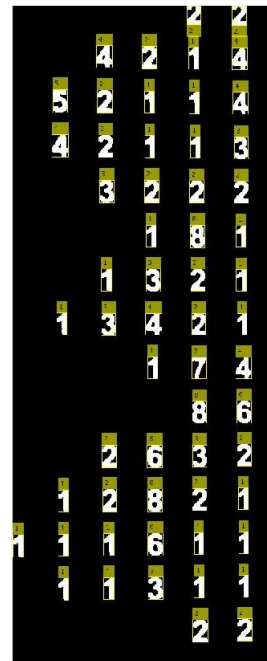


Figure 5: Recognised digits

5 CONCLUSION

Knowing the solution, displaying the correct one is a simple task. It can be achieved by overlaying the solution over the original image, or by creating a GUI (graphical user interface) in matlab, in which the user can ask, whether the selected cell should be filled or left empty.

REFERENCES

- [1] NIXON, Mark S. a Alberto S. AGUADO. Feature Extraction and Image Processing. 2008. London, UK: Elsevier, 2002. ISBN 978-0-12372-538-7.
- [2] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. Image Processing, Analysis, and Machine Vision. 2008. Toronto, Canada: Thomson Learning, 2008. ISBN 978-0-495-24428-7.